Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

Hardware	David Anear
Software	Jeff Rae
Amateur Radio	Clive Harman VK3BUS
Education	Jeff Kerry
Library	Ron Kerry
Tape Library	John Whitehead
Disk Library	David Dodds (B.H.)
Newsletter	Ian Eyles
Sym	Brian Campbell
Secretary	Rosemary Eyles

OSI SYM KIM AIM UK101 RABBLE 65

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.4 No.9 June 1984

You will have to read slowly this month, only 12 pages of your favourite newsletter. Murphy's Law again. It was a combination of Ian being sick, the typewriter playi.g up and a long weekend. Anyone who has an IBM typewriter will know of the trouble that can be caused by the rotate band. Hopefully, things will be back to normal next month.

The Meeting Was KAOS has been omitted because of space, as not a lot happened at the last meeting, except Ray Gardiner's talk about America, (perhaps we will be able to get him to do an article about his trip for the next newsletter?). There was one item we think should be mentioned. Andrew Stephanou has heard from a few BBC owners and he is in contact with two interstate user groups, so he is hoping to have more to report soon.

We have been offered space in the user group section at the 3rd Australian Personal Computer Show which we have accepted. This show runs from 18-21 July 1984 and we are looking for volunteers to help man the stand, one of the obligations of accepting this offer is that the stand must be manned while the show is open (9am-7pm), so if you can help please contact us. We shall also be getting reduced price admission tickets for the show which will allow our members to visit the show for $3 instead of $4. Please let us know if you are interested.

The next meeting will be at 2pm on Sunday 24th June at the Essendon Primary School which is on the corner of Raleigh and Nicholson Streets, Essendon. Would the usual early arrivers please note that the children will be there at 10am. The school will open for our normal meeting at 1pm.

The closing date for items for the next issue is 13th July.

INDEX

WHAT YOU ALWAYS WANTED TO KNOW ABOUT THE COMPSOFT CP/M SYSTEM AND GOT AROUND TO ASKING

by Michael Lemaire

We've had a few queries regarding our Rabble/Ohio CP/M system, kindly relayed to us by Rosemary and Ian, and so here unfolds a glimmer of light upon various mysterious magickes.

Q. Is the system like a Big Board, only using the Rabble/Ohio as a terminal?
A. No. Since the Rabble and Ohio have circuitry for screen, keyboard, printer, and serial I/O, it seemed a waste of board space, component and money to provide these features on the Z80 card. The card has enough circuitry to communicate with the host computer, ie. the Rabble or Ohio, and the host controls all I/O devices, including the FDC card. This means (a) the Z80 BIOS is pretty small, leaving more usable memory, and (b) we can do some fancy things at the host computer end, like printer spooling and disk cache.

Q. Is there provision for a serial or parallel printer interface on the CP/M card?
A. No, because the necessary hardware was already available on the host machine.

Q. Would the board work with a terminal?
A. No, because the CP/M system was designed to work under a host computer. Of course, you could always connect a terminal to the host.

Q. How are Ohio users supposed to use programs such as WordStar without 80 columns?
A. The Ohio system manages an 80 by 24 VIRTUAL screen, of which 64 columns can be shown at any one time. The user can display either the left three-quarters of the screen (columns 0 to 63), or the right three-quarters (16 to 79). The screen can either be manually switched CONTROL -< for the left window, and CONTROL -> for the right window), or the user can set the console to switch automatically.
(NB. WordStar can be configured for 64 column screens if you really want to.)

Q. Is there provision for a serial interface for inter-computer CP/M transfers?
A. The host machine has a serial interface, and can be used under CP/M as the PUN (output) and RDR (input) devices, as usual.

Q. Why are the CP/M and FDC boards only available fully built?
A. Because if we build them, they will work, and we won't have to waste our time fixing other peoples problems. I've lost count of the number of Tasan Video boards, Rabble expansion boards and Rabble 65 boards that we've had to look at.

Q. Wouldn't it be easier and cheaper to pull out the 6502 and plug in a Z80 on my Ohio?
A. Cheaper, most certainly. Easier, certainly not. Firstly, you won't be able to run much CP/M stuff on a 48K system (you can forget about dBase II, WordStar and most compilers), so you'll need to add 16K RAM to get a 64K system. That means that all the I/O will have to be bank-selected. You'll need to write your own BIOS to suit the system - not hard, but it's involved. The biggest problem will be getting software on Ohio's disk format, which is

somewhat unique and unpopular. You'll need a friend with a standard disk CP/M computer and communications software.

Q. I'd ike to put CP/M on my Ohio, but how can I get the Full ASCII character set for eg. {} in Pascal and C?
A. The Ohio system software supplied with the CP/M cards provides a full ASCII keyboard driver, with proper CAPS LOCK operation. Every printable ASCII character is directly available from the keyboard.

If you have any queries concerning the Compsoft CP/M system, we can be contacted at Compsoft, 235 Swan St, Richmond, 3121, ph 03 429 9686, or through KAOS.

KAOS - W.A.

Our last meeting was well attended with 13 people present. We also had two new members. It seems that many of our so called "ORPHAN" computers are gaining "new parents" who need the back-up and support provided by an active users group.

As usual, a lot of discussion and sharing of ideas took place.

The next meeting is on Sunday July 21st at 2pm back at our usual meeting place. Guild House (top floor), 56 Kishorn Rd, Mt Pleasant.
Gerry Ligtermoet,

FOR SALE

Tasker Bus RAM board, 24K CMOS uses 12 of Hitachi HM6116P-3, Assembled and tested. $70.00
MPI B51 Disk Drive and Tasker Bus controller board. $180.00
Contact Frank Brown

I.C.L. Termiprinter. Daisy belt type, operates at 110, 300 or 600 baud. Accepts RS-232 signals ex C1P. Prints up to 118 chrs wide on standard 11" x 12 7/8" pin feed paper at 10 chrs per inch. Standard ASCII chrs. Complete with handbook. $200.00
Roger Haigh,

C1P Series II complete with fully populated Rabbleboard. Screen enhancement Board gives 20 screen formats. ROMs include Cegmon, Toolkit, Exmon. Extensions to BASIC have been merged with the language. Has been running with 5.25" disk. Centronics printer port and cable, and disk cable included, but no drive. Includes Hexdos, 65D, and assorted software, plus full documentation. $450.00
Paul Brodie,

Superboard

©June, 1984.

NEWSLETTER OF THE OHIO SUPERBOARD USER GROUP, 146 YORK STREET, NUNDAH, 4012.

FASTSEARCH

Ever written things sequentially in a notebook, and then had to laboriously search through the pages, unsure if the item you want is even there?

In my constant quest for supremacy over the bookmakers, I kept such a book. I decided to use the computer to store my notes - a memory that never forgets, and a much faster reader than me!

The type of information I wanted was current, no more than two months old. Older data than this could be deleted each week, the lines renumbered, and new data added. Every Friday night I would type in the names of over a hundred horses, and use the program to find any information that I had noted about them in the preceding eight weeks. Such a program is not really suited to disk operation because of the time delay in a search, plus the considerable wear and tear on disks during the hundred odd separate searches. The length of the program varies around the 20k mark.

The Basic program I used to search was as fast as I could get it, but the delay was still annoying. The solution is the M/C routine which follows. The results are as you would expect. Information retrieval is virtually instant. The routine will print anything following the search string to the end of the line. A Basic line can contain the string in a DATA or REM statement, or even just the string itself so long as none of Basic's reserved words or symbols are used. The actual search string is not printed unless it appears again within the line.

Basic Driver Routine:

```
10 POKE 11,53:POKE 12,2:REM FASTSEARCH
20 INPUT"Name of subject";A$:X=USR(X):PRINT:GOTO 20
```

Fastsearch M/C:

```
0235  A9 03             LDA #$03      Set up program start address each
0237  A0 00             LDY #$00      time routine is called. End of
0239  85 59             STA  $59      input buffer is used as storage.
023B  84 58             STY  $58
023D  20 6C A8 CRLF     JSR  $A86C    Do a carriage return/line feed. Note
0240  A2 00    START    LDX #$00      no effect on register contents.
0242  20 68 02 NEXT     JSR  MEMINC   Search address is in $0058,59
0245  A5 58             LDA  $58      Check for end of Basic program
0247  C5 7B             CMP  $7B      storage, first lo byte, then hi byte.
0249  D0 06             BNE  KEEPGO
024B  A5 59             LDA  $59
024D  C5 7C             CMP  $7C
024F  B0 1D             BCS  RETURN   If end, return to Basic.
0251  B5 13    KEEPGO   LDA  $13,X    Get character from input buffer.
0253  F0 07             BEQ  PRINT    If zero, we have a matched string.
0255  D1 58             CMP ($58),Y   Compare with memory character
0257  D0 E7             BNE  START    No match, reset & keep looking.
0259  E8                INX
025A  D0 E6             BNE  NEXT     Check next character (branch always.)
025C  B1 58    PRINT    LDA ($58),Y   Get a character from memory.
```

```
025E  F0 DD               BEQ  CRLF    If zero, end of Basic line.
0260  20 68 02            JSR  MEMINC  Get ready for another character.
0263  20 EE FF            JSR $FFEE    Print character.
0266  D0 F4               BNE  PRINT   Branch always.
0268  E6 58     MEMINC    INC $58      Lo byte.
026A  D0 02               BNE  RETURN
026C  E6 59               INC $59      Hi byte.
026E  60       RETURN     RTS
```

The routine should work on any Basic-in-Rom OSI. For 65D users, input buffer is from $001B to $0062. End of program address is in $007A,7B. Location for storage will have to be end of memory. Pointer is at 8960 decimal. USR vectors at 8955 ,8956, decimal.(lo,hi bytes).I am not sure where the CRLF routine or screen driver lives? It would be easy to incorporate a CRLF routine in the M/C program.

- -

A BBC REPORT by Robin Wells.

After four years of owning an OSI, building it up from a 600 board to a twin disk system with all the Roms and software available, I decided to update to something more modern. Like David Anear, I looked at what was available, rejecting the overpriced and non-graphics machines running CP/M. I considered the Commodore 64 and Apple clones, but these machines do not offer much of an advance on OSI. Though good graphics can be obtained from both, the antique versions of Basic make the programming unnecessarily difficult.

When I had to make a trip to the UK recently, I came away with a BBC/B model with disk interface. The total bill was just under A$600. Acorn has opened a factory in Hong Kong to supply the Asian market, and I would expect this to have the effect of reducing the prices.

The BBC runs a 6502A at 2 MHz. The A model supports 16k of Ram, and has four display modes. Upgrading to a B model gives 32k Ram, and eight display modes.

640 x 256	2	colour graphics	80 x 32		
320 x 256	4	" "	40 x 32		
160 x 256	16	" "	20 x 32		
" "	2	" "	80 x 25		
320 x 256	2	" "	40 x 32	*	
160 x 256	4	" "	20 x 32	*	*=Model A
40 x 25	2	Colour Text		*	
40 x 25		Teletext Display		*	

Both models have the same Rom capability. Five sockets are provided inside, two being for the Basic interpreter and the machine operating system, and in the case of the Model B, a third socket is occupied by the disk system. Many Rom expansion boards are on the market to cater for the large number of Roms that have become available to fill those two sockets. All up, the BBC can have resident up to 256k of Rom. Acorn engineers have come up with an ingenious method whereby utility Roms can be called up by either their full or a shortened name, and the BBC finds the appropriate Rom, and places it in the memory map. Magazines like Micro User carry advertisements to encourage you to attempt to fill up that huge addressable memory space.

Fitting the ex-OSI drives (MPI B51) proved to be easy. I simply removed the data separators and connected them to the disk ribbon cable socket under the machine. There are four other such sockets, wired for Printer, User I/O port, 1 MHz BUS, and the TUBE (for connecting a second processor).

At the back of the BBC are connectors for UHF Output, Composite Video, R-G-B Video, RS423, Cassette storage, Analogue input (Joysticks or control uses), and Econet.

Econet is designed for education use in a classroom situation, and enables the machine to be networked easily to up to 254 other BBCs. The BBC also boasts a three channel sound generator with separate envelope control, complete with internal amplifier and speaker. It is capable of spectacular sound effects.

In a future article, I'll go into the BBC in more depth. If other BBC users would like to contact me, the address is

---

MORE ON NOS BASICODE

KAOS 4/6 contained a program for the OSI to provide the NOS BASICODE format on tape. To use the program, enter it using the M/C monitor at .1E00 G. Some housekeeping is done to set up the USR vectors, and to protect the code from being overwritten by Basic programs and strings, before exiting to a warmstart. To use the program, you call it with X=USR(X).

The program prompts you with S/L/R ? S and L are SAVE and LOAD. You can hit L anytime as the program ignores noise and rubbish on the tape. The program starts saving immediately you hit S, so start the tape beforehand. The R command simply removes the BASICODE translation program by resetting the top of memory pointers. The BASICODE program can be relocated to higher memory using EXMONs Relocate function, and entering some further changes manually.

BASICODE Specifications:

Two tones are used to record the data onto cassette tape, with frequencies of 1200 and 2400 Hz. A Ø is defined as one full cycle of 1200 Hz, and a 1 is two full cycles of 2400 Hz. The Baud rate is 1200 Baud. A byte on tape is represented by 1 start bit (Ø),8 data bits(LSB first), and two stop bits(1).

The Basic program is coded either in the form in which it has been typed in, or as it is displayed when the LIST command is given. All letters are presented in ASCII. Each Basic Instruction should be followed by a space. Each Basic line is closed with a carriage return. All ASCIIs receive a closing bit = 1.

The tone sequence on the tape is: ASCII start text($82), Basic information in ASCII, ASCII end of text ($83), Checksum.

The checksum is the result of the bit indication exclusive-or from all previous bytes. The checksum is an 8 bit term, and in this case, the 8th bit can be a Ø. The checksum is intended to allow the user to check whether the program has been read without errors. The program has been loaded, however, errors or not.

BASICODE Protocol:

The use of BASICODE requires more rules than may be apparent at first glance. Not only do we wish to transfer a listing of a program from a computer type A to a computer type B, but ideally it should also be possible to execute the program on both computers. This will not be possible in all instances, so the program should be written so that it is easy to understand and hence modify.

The OSI Basic is a standard 8k Microsoft Basic. The commands to avoid are PEEK, POKE, DEF FN, and USR.

Computer screens, rows and columns, vary enormously. This is not so terribly important, but the number of ciphers per line is very important. The number of characters in the program line should not exceed 60 characters including the line number, and preferably only one statement per line number.

To be continued. . . . .

Next Month, we conclude the BASICODE article. Some changes for the CEGMON monitor. Adventuring part 2.

Ed Richardson.

APPLE DISK SYSTEM

by David Anear

In June 1978 when Apple released its first disk systems using 5.25 disk drives, there were no standards prevailing at the time. OHIO had an 8" system on the market and firms like SWTP had no disk systems as yet. Most disk controllers were TTL based and cost more than the hobbyist could afford. Apple's disk system was like Ohio's - cheap and reliable, but unlike Ohio it was designed for a 5.25" low cost mechanism. Ohio later released its 5.25" system, which was a patched 8" DOS running on a MPI drive.

The Apple system used the slow but reliable Shugart mechanism. The mechanism has no index led, and no track zero swit h and only consists of the disk drive motor,write protect switch and a read/write head with its stepper motor. The normal electronics board was replaced by a very simple analog board containing a set of buffers, a transistor array and a op-amp package for reading from the head. The electronics board does not even have any logic to drive the stepper motor, but relies on the Apple itself to do this chore.

By comparison the Ohio disk drives are a standard off the shelf MPI unit using index, track zero leds and a high speed head positioner with an electronics board using 20+ ICs.

Most computers nowadays use a sophisticated disk controller based around one of the Western Digital controller chips and use IBM type formatting. Neither Apple or Ohio used these chips as they were not available till well after their disk systems were released, and they are expensive. Ohio used a PIA to control the disk, and a ACIA to read/write the data and a few house keeping chips to do the encoding as well as a 125 kcs clock. The boot strap loading program is in the system monitor.

Apple's controller consists of a pair of fuse link proms holding the boot program, a data latch and a timer chip to switch on the drive motor. It's even simpler than the Ohio controller. The big difference between Apple disks and anyone elses is the way the data is packed onto the disk. Ohio disks are not easily read by a IBM type machine due to the use of a ACIA to convert the data, which adds extra stop bits to the stream. Apple disks use no start/stop bits and have an unusual encoding system as well as self syncing bytes (no index led remember) to locate start of track/sectors.

This non use of the index hole means you can flip a normal single sided disk over and use the other side if you punch a new write protect notch. The upshot of all this is that an Apple single sided disk holds as much data as a double density IBM formatted disk or about twice a comparable Ohio disk.
Here is a table to compare Ohio and Apple disk formatting:-

	*	OHIO 65D 3.2	*	APPLE 3.3 DOS	*
TRACKS	*	40	*	35	*
SECTORS PER TRACK	*	8	*	16	*
SECTORS PER DISKETTE	*	320	*	560	*
BYTES PER SECTOR	*	256	*	256	*
BYTES PER DISKETTE	*	81,920	*	143,360	*
OVERHEADS (DOS, DIR ETC)	*	14,080	*	15,360	*
USABLE BYTES PER DISK	*	67,840	*	128,000	*
SMALLEST USABLE AREA (M.CODE)	*	1 sec	*	1 sec	*
SMALLEST USABLE AREA (BASIC)	*	8 sec	*	1 sec	*
MAX NUMBER FILES IN DIR	*	40	*	105	*
TIME TO LOAD 20K BASIC PROG	*	4-5 sec	*	19 sec	*

As you can see from the table Apple DOS is slower than Ohio's but with the new fast DOS systems available for the Apple, the times are very close.

A final note:- Apple's new Mackintosh computer uses the same type of controller as the Apple][but it has been put onto a single chip and with the Sony 3.50" drive has a storage capacity of 400 kbytes; compare this to the IBM personal computer which stores 145 kbytes and has more chips in its floppy disk controller than the complete Apple Mackintosh!!
Next month - Apple Graphics

PRINTER PLOTTER
by Gerard Campbell

If you have an 8K Superboard and a whatever-80 printer or similar then you can obtain a hardcopy of a 'virtual' or invisible screen created above BASIC workspace. This 'screen's' size is limited only by available memory and page size. In this instance I chose a resolution of 160 horizontal X 96 vertical dots requiring 1920 bytes (bit mapped-8 horz'l bits = one memory byte), starting at $1800 (6144).

Listing 1 does the following:-

1. Line 10 sets BASIC workspace top to $1800. Line 220 clears the virtual screen.
2. Subroutine 100 plots a point. = PLOT X,Y
3. Subroutine 3000 draws a line from X1,Y1 to X1,X2. = DRAW X1,Y1 TO X2,Y2
4. Subroutine 4000 is a bit image graphics dump of the virtual screen.
   Lines 4005 and 4020 set up the printer. (A CP-80)
5. Subroutine 5000 produces a large plot using condensed,
   half character blocks. Line 5005 sets up printer.

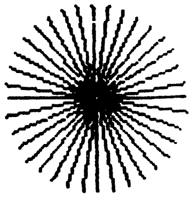
Listings 2-7 are calculatory programs used with listing 1 to produce the plots shown. Plots are printed with the Y-axis across the page with the last column printed first. All examples take a few minutes to plot but don't go to sleep!!

This program should be easy to modify to work on any computer with some spare memory and a whatever-80 printer including the RABBLE 65.

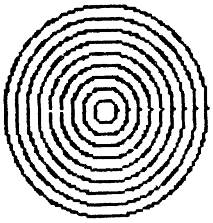
LISTing 1

```
 10 POKE 133,00:POKE 134,24:GOTO200
 20 :
 90 REM .....PLOT X,Y  Subroutine
 100 IF X<0 OR X>159 THEN RETURN
 102 IF Y<0 OR Y>95 THEN RETURN
 105 XB=INT(X/8):RM=X-XB*8:PB=ST+XB+Y*20
 110 CV=PEEK(PB):VO=2^RM:POKE PB,(CV OR VO)
 120 RETURN
 125 :
 200 VS=1920:ST=6144:DIM X(160),Y(160)
 220 FOR I=0 TO VS:POKE ST+I,0:NEXTI
 240 :
 300 REM ....INSERT CALCULATION PROGRAM HERE
 310 REM ....SEE EXAMPLES ELESWHERE
 320 :
 2800 END
 3000 REM .........DRAW X1,Y1 TO X2,Y2
 3010 NB=ABS(X2-X1):SB=SGN(X2-X1)
 3020 IF NB>1 THEN 3030
 3025 X(0)=X1:Y(0)=Y1:X(1)=X2:Y(1)=Y2:NB=1
 3026 GOTO 3200
 3030 MX=(Y2-Y1)/(X2-X1)
 3040 FOR Q=0 TO NB:X(Q)=X1+Q*SB
 3045 Y(Q)=Y1+INT(Q*MX)*SB
 3050 X=X(Q):Y=Y(Q):GOSUB100:NEXTQ
 3200 FOR Q=0 TO NB-1
 3210 IF ABS(Y(Q+1)-Y(Q))<2 THEN 3250
 3220 NU=SGN(Y(Q+1)-Y(Q)):X=X(Q)
 3230 FOR K=Y(Q)+NU TO Y(Q+1)-NU STEP NU
 3240 Y=K:GOSUB 100 PLOT X,Y :NEXTK
 3250 NEXT Q :RETURN
 3260 :
 3990 REM .....BIT IMAGE PLOT
 4000 WP=ST+19:WM=WP:POKE 15,120
 4001 SAVE
 4005 PRINT CHR$(27);"A";CHR$(7);
 4010 FOR T=1 TO 20
 4020 PRINT CHR$(27);"K";CHR$(96);CHR$(0);
 4030 FOR N=1 TO 96:CV=PEEK(WP)
 4065 IF CV=13 OR CV=127 THEN CV=CV-1
 4080 PRINT CHR$(CV); :WP=WP+20:NEXT N
 4090 WP=WM-1:WM=WM-1:PRINT:NEXTT
 4150 POKE 15,72:RETURN
 4200 :
 4990 REM ....COMPRESSED BLOCK CHARACTER PLOT
 5000 WP=ST+19:WM=WP:POKE 15,120
 5001 SAVE
 5005 PRINT CHR$(27);"A";CHR$(4);CHR$(15);
 5010 FOR T=1TO20:FOR S=7 TO 0 STEP -1
 5020 FOR N=1TO96:CV=PEEK(WP):AV=INT(2^S)
 5025 J=CV AND AV
 5030 IF J=0 THEN PRINT " ";
 5040 IF J=AV THEN PRINT CHR$(4);
 5050 WP=WP+20:NEXT N:WP=WM:PRINT:NEXTS
 5060 WP=WM-1:WM=WM-1:NEXTT
 5070 POKE 15,72:RETURN
OK
```

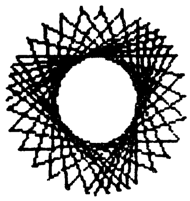
```
290 REM .......Listing 2.
300 X1=80:Y1=40
310 FOR AN=0 TO 360 STEP 10
320 X2=80+INT(40*SIN(AN*0.0175))
330 Y2=40+INT(40*COS(AN*0.0175))
340 GOSUB 3000 :NEXT AN :GOSUB 4000
350 END
```



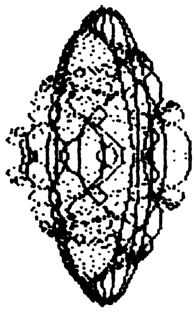
```
290 REM ......Listing 3.
300 FOR D=0 TO 8
305 FOR AN=0 TO 360
310 X=INT((45-(D*5))*SIN(AN*0.0175))
320 Y=INT((45-(D*5))*COS(AN*0.0175))
330 X=X+80:Y=Y+45
340 GOSUB 100:NEXTAN:NEXTD:GOSUB4000
350 END
```



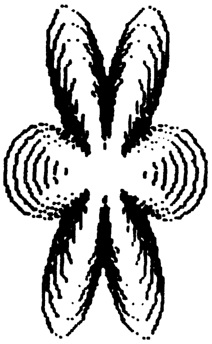
```
290 REM ........Listing 4.
300 E=40:X1=E+90:Y1=45
310 FOR AN=0 TO 8000 STEP 235
320 X2=INT(90+E*COS(AN*0.0175))
330 Y2=INT(45+E*SIN(AN*0.0175))
340 GOSUB3000
350 X1=X2:Y1=Y2:NEXTAN:GOSUB4000
360 END
```



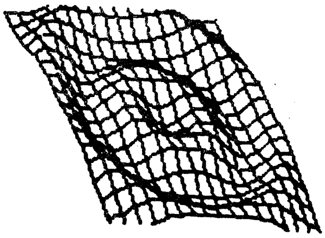
```
 290 REM .......Listing 5.
 300 U1=64:U2=U1*U1:V1=48:V2=V1
 310 FOR T=0 TO U1: U4=T*T: A=SQR(U2-U4)
 320 FOR I=-A TO A STEP 6
 330 R=(SQR(U4+I*I))/64
 340 F=(R-1)*COS(18*R) : Y=I/3+F*V2
 350 IF I=-A THEN M=Y:M1=Y:Y=V1+Y:GOTO 400
 360 IF Y<=M THEN 430
 370 M=Y:Y=V1+Y
 400 X=INT(U1-T+10):Y=INT(Y):GOSUB100
 410 X=INT(U1+T+10):Y=INT(Y):GOSUB100
 420 NEXT I: NEXT T
 425 GOSUB 4000 :END
 430 IF Y>M THEN 420
 440 M1=Y:Y=V1+Y:GOTO 400
OK
```



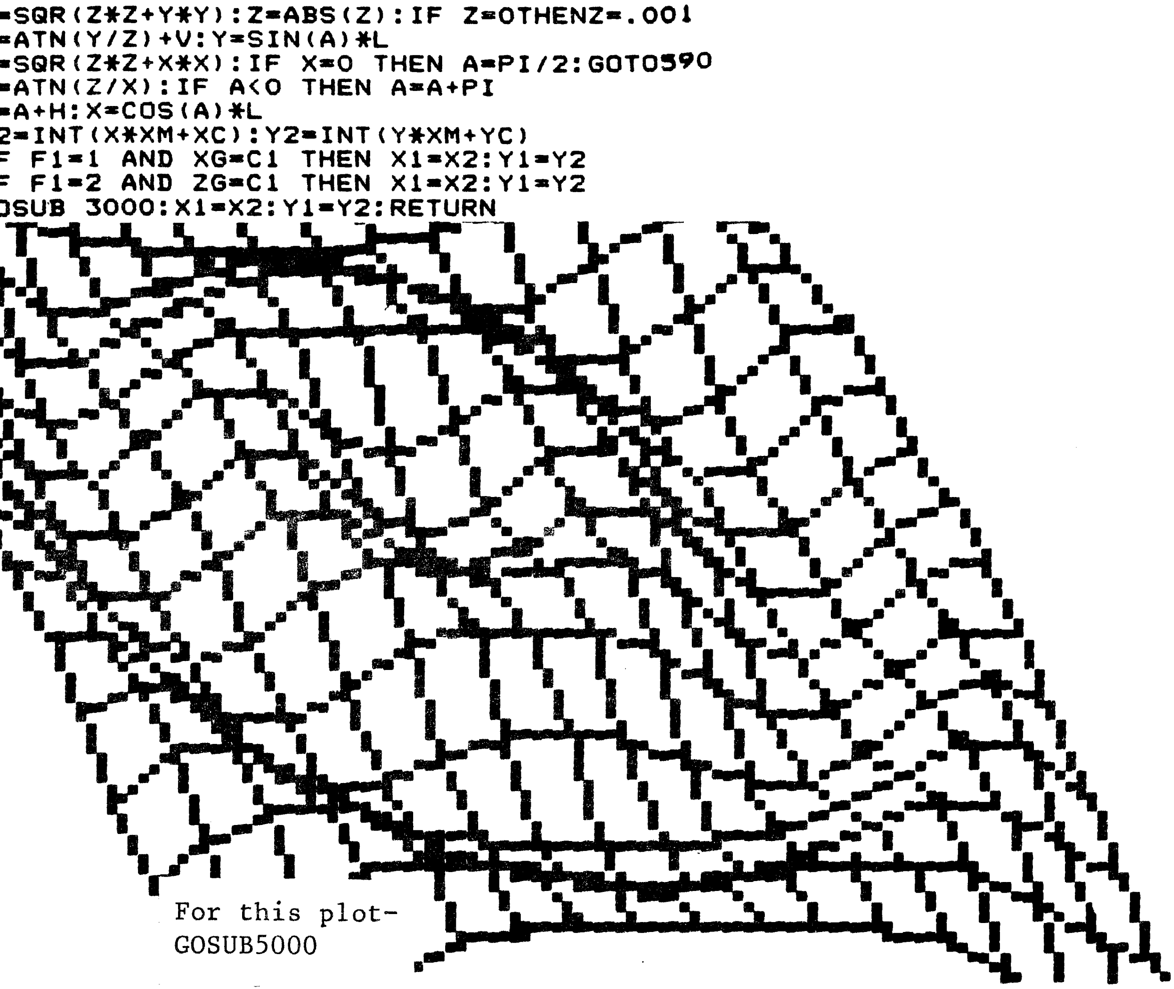
```
290 REM ........Listing 6.
300 PI=3.142:P2=2*PI:E=P2/180:N1=5:N2=4
305 XC=80:YC=48:XM=160/180:YM=90/180
310 LE=3:FOR D=1 TO 10
320 M1=N1*D:M2=N2*D
330 FOR AN=0 TO P2 STEP E:A=AN
340 SN=ABS(SIN(AN*LE)):RA=SN*M1+M2
350 X=COS(A)*RA:Y=SIN(A)*RA
360 X2=INT(X*XM+XC):Y2=95-INT(Y*YM+YC)
370 IF AN=0 THEN X1=INT(X2):Y1=INT(Y2):GOTO400
380 X2=INT(X2):Y2=INT(Y2):GOSUB3000
390 X1=X2:Y1=Y2
400 NEXT AN
410 GOSUB 3000:NEXTD
415 GOSUB 4000
420 END
```



```
280 REM ......Listing 7.
290 PI=3.142:P2=2*PI:E=15:MF=100/(PI*2.5):F1=1
300 XC=80:YC=48:XM=.44:YM=.35
310 HT=30*PI/180:VT=63*PI/180:C1=-120:C2=120
340 FOR ZG=C1 TO C2 STEP E
350 FOR XG=C1 TO C2 STEP E
360 X=XG:Z=ZG:L=SQR(Z*Z+X*X):L=L/MF:Y=COS(L)*25:GOSUB 530
400 NEXT XG:NEXT ZG
410 F1=2
420 FOR XG=C1 TO C2 STEP E
440 FOR ZG=C1 TO C2 STEP E
450 X=XG:Z=ZG:L=SQR(Z*Z+X*X):L=L/MF:Y=COS(L)*25:GOSUB 530
460 NEXT ZG:NEXTXG
500 GOSUB 4000:END
530 IF ZG<0 THEN V=VT:H=-HT:GOTO550
540 V=-VT:H=HT
550 L=SQR(Z*Z+Y*Y):Z=ABS(Z):IF Z=0THENZ=.001
560 A=ATN(Y/Z)+V:Y=SIN(A)*L
570 L=SQR(Z*Z+X*X):IF X=0 THEN A=PI/2:GOTO590
580 A=ATN(Z/X):IF A<0 THEN A=A+PI
590 A=A+H:X=COS(A)*L
600 X2=INT(X*XM+XC):Y2=INT(Y*XM+YC)
610 IF F1=1 AND XG=C1 THEN X1=X2:Y1=Y2
620 IF F1=2 AND ZG=C1 THEN X1=X2:Y1=Y2
630 GOSUB 3000:X1=X2:Y1=Y2:RETURN
```



For this plot-
GOSUB4000



For this plot-
GOSUB5000

OS65U SINGLE-DISK COPIER

by Leo Jankowski

Lines 80 to 250 are based on a listing of D. Stevenson's; from the March NL of the Tristate U.G. Lines 340 to 480 come from OSI TI1017 - 10.14.81, found in my OS65U manual. The rest is mine! The program copies blocks of 6*3584 bytes at a time, assuming that there is 48K of RAM.

NB = # of bytes to copy.
RA = first RAM address of Copy Buffer.

It is necessary to first INIT the disk and copy Track Zero with the System Copy Program. Edit this Copy Program by inserting the lines:

```
1695  PRINT:INPUT"Insert MASTER disk ";X$
1865  PRINT:INPUT"Insert DESTINATION disk ";X$
```

The two line numbers above may have to be amended for your Copy Program.

If the Copy Program can be stopped in this way for single disk copying, then why not carry on and copy the whole disk? Because the OS65U Copy Program will only copy 3584 bytes at a time. My program copies upto 21504 bytes at a time.

```
10 REM OS65U SINGLE DISK COPIER BY - LZJ
20 FORX=1TO32:PRINT:NEXT:FORX=1TO40:PRINTTAB(10)"-";:NEXT:PRINT
30 POKE2888,0:PRINTTAB(10)":SINGLE DISK DRIVE COPIER for 48K RAM :"
40 FORX=1TO40:PRINTTAB(10)"-";:NEXT:FORX=1TO4:PRINT:NEXT
50 POKE132,255:POKE133,107:CLEAR:RA=108*256:NB=6*3584:CB=9889:P=256
60 PRINTTAB(8)"Use COPIER utility to INIT & to copy TRACK 0."
70 DV$="A":FORX=1TO7:PRINT:NEXT
80 PRINT"You can copy between any TWO memory limits, (except TRACK 0)
90 PRINT:PRINT"They will be rounded to the next page."
100 PRINT:INPUT"Start address ";DA:PRINT
110 IFDA<3584THENDA=3584
120 DA=INT(DA/P)*P:TR=INT(DA/3584)
130 PRINT"That rounds to "DA"(Track "TR", Page"(DA-TR*3584)/P")"
140 PRINT:INPUT"Is this OK (Y/N) ";A$
150 IFA$="ABORT"THENPRINT:PRINT"ABORTED":GOTO470
160 IFLEFT$(A$,1)<>"Y"THEN100
170 PRINT:INPUT"End Address (plus 1) ";EN
180 IFEN>275968THEN EN=275968
190 IFEN<>(EN/P)*PTHENEN=1+INT(EN/P)*P
200 TR=INT(EN/3584)
210 PRINT:PRINT"That rounds to "EN"(Track "TR", Page"(EN-TR*3584)/P")"
220 PRINT:INPUT"Is this OK  (Y/N)  ";A$
230 IFA$="ABORT"THENPRINT:PRINT"ABORTED":GOTO470
240 IFLEFT$(A$,1)<>"Y"THEN170
250 IF DA=>ENTHENFORX=1TO30:PRINT:NEXT:GOTO100
260 GOSUB340
270 IFEN-DA<NBTHENNB=EN-DA
280 PRINT:INPUT"Insert MASTER disk, hit <RETURN> ";Q$:RW=0:GOSUB390
290 PRINT:INPUT"Insert COPY disk, hit <RETURN> ";Q$:RW=1:GOSUB390
300 PRINT:PRINT"Copied: "DA"->"DA+NB-1:PRINT:PRINT
310 DA=DA+NB:IFDA=ENTHENPRINT:PRINT"COPY COMPLETED!":GOTO470
320 GOTO270
330 :
340 POKE8778,192:POKE8779,36:POKE9432,243:POKE9433,40
350 POKE9435,232:POKE9436,40
360 POKECB+7,RA-INT(RA/P)*P:POKECB+8,INT(RA/P)
370 RETURN
380 :
390 POKECB+5,NB-INT(NB/P)*P:POKECB+6,INT(NB/P)
400 DH=INT(DA/16777216):RM=DA-DH*16777216
410 DM=INT(RM/65536)    :RM=RM-DM*65536
420 DL=INT(RM/P)        :RM=RM-DL*P
430 POKECB+1,RM:POKECB+2,DL:POKECB+3,DM:POKECB+4,DH
440 DEV DV$:ER=USR(RW)
450 IFER=0THENRETURN
460 PRINT:PRINT"Error ";ER;" at Address ";DA
470 POKE8778,208:POKE8779,16
480 POKE132,0:POKE133,192
```